



US008321371B2

(12) **United States Patent**
Bridges et al.

(10) **Patent No.:** **US 8,321,371 B2**
(45) **Date of Patent:** ***Nov. 27, 2012**

(54) **ENHANCED ARTIFICIAL INTELLIGENCE LANGUAGE**

(75) Inventors: **Matthew Bridges**, Newton, MA (US);
Raymond C. Kurzweil, Newton, MA (US)

(73) Assignee: **Kurzweil Technologies, Inc.**, Wellesley Hills, MA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1191 days.

This patent is subject to a terminal disclaimer.

6,247,007	B1	6/2001	Cooperman et al.	
6,505,162	B1 *	1/2003	Wang et al.	704/275
6,615,111	B2	9/2003	Tackett et al.	
6,751,606	B1	6/2004	Fries et al.	
7,152,092	B2 *	12/2006	Beams et al.	709/204
7,257,537	B2 *	8/2007	Ross et al.	704/270
7,305,372	B2 *	12/2007	Bridges et al.	706/45
7,337,157	B2 *	2/2008	Bridges et al.	706/45
7,437,292	B2 *	10/2008	Feldes et al.	704/270
7,539,656	B2 *	5/2009	Fratkina et al.	706/45
7,610,556	B2 *	10/2009	Guo et al.	715/745
7,707,135	B2 *	4/2010	Bridges et al.	706/62
2002/0038213	A1 *	3/2002	Adachi	704/257
2003/0055623	A1 *	3/2003	Epstein et al.	704/1
2003/0187648	A1 *	10/2003	Dharanipragada et al.	704/260
2003/0191627	A1	10/2003	Au	
2010/0205140	A1 *	8/2010	Bridges et al.	706/54

(21) Appl. No.: **11/949,883**

(22) Filed: **Dec. 4, 2007**

(65) **Prior Publication Data**

US 2008/0218473 A1 Sep. 11, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/379,414, filed on Mar. 4, 2003, now Pat. No. 7,305,372.

(51) **Int. Cl.**

G06F 15/00 (2006.01)

G06F 15/18 (2006.01)

(52) **U.S. Cl.** **706/62; 706/45; 706/47; 706/52; 706/55; 704/1; 704/3; 704/4; 704/200; 707/913; 707/917**

(58) **Field of Classification Search** None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,222,188	A *	6/1993	Hutchins	704/200
5,806,060	A	9/1998	Borgida et al.	

OTHER PUBLICATIONS

Ringate et al., "AIML Reference Manual" <http://www.alicebot.org/documentation/aiml-reference.html> (2001), 31 pages.

Ringate et al., "AIML Primer" <http://www.alicebot.org/documentation/aiml-primer.html> (2002), 8 pages.

Wallace, "Symbolic Reductions in AIML" <http://www.alicebot.org/documentation/srai.html> (2000), 5 pages.

"A.L.I.C.E. AI Foundation" <http://www.alicebot.org/alice/aiml.html>, 2 pages.

Wallace, "AIML Overview" <http://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html>, 7 pages.

* cited by examiner

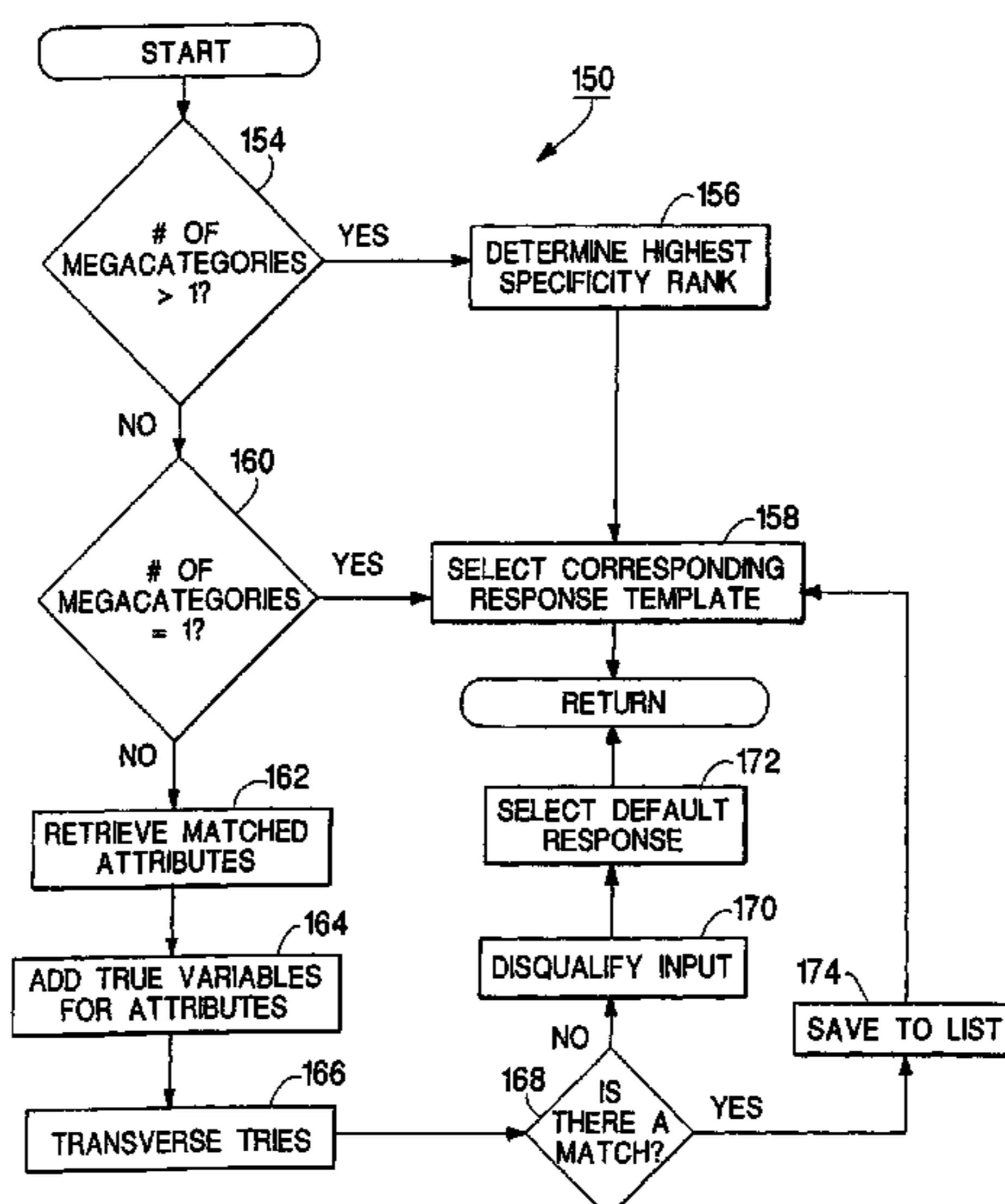
Primary Examiner — Omar Fernandez Rivas

(74) *Attorney, Agent, or Firm* — Fish & Richardson P.C.

(57) **ABSTRACT**

A method of determining an appropriate response to an input includes linking a plurality of attributes to a plurality of response templates using a plurality of Boolean expressions. Each attribute is associated with a set of patterns. Each pattern within the set of patterns is equivalent. The method also includes determining an appropriate response template from the plurality of response templates based on the input.

55 Claims, 4 Drawing Sheets



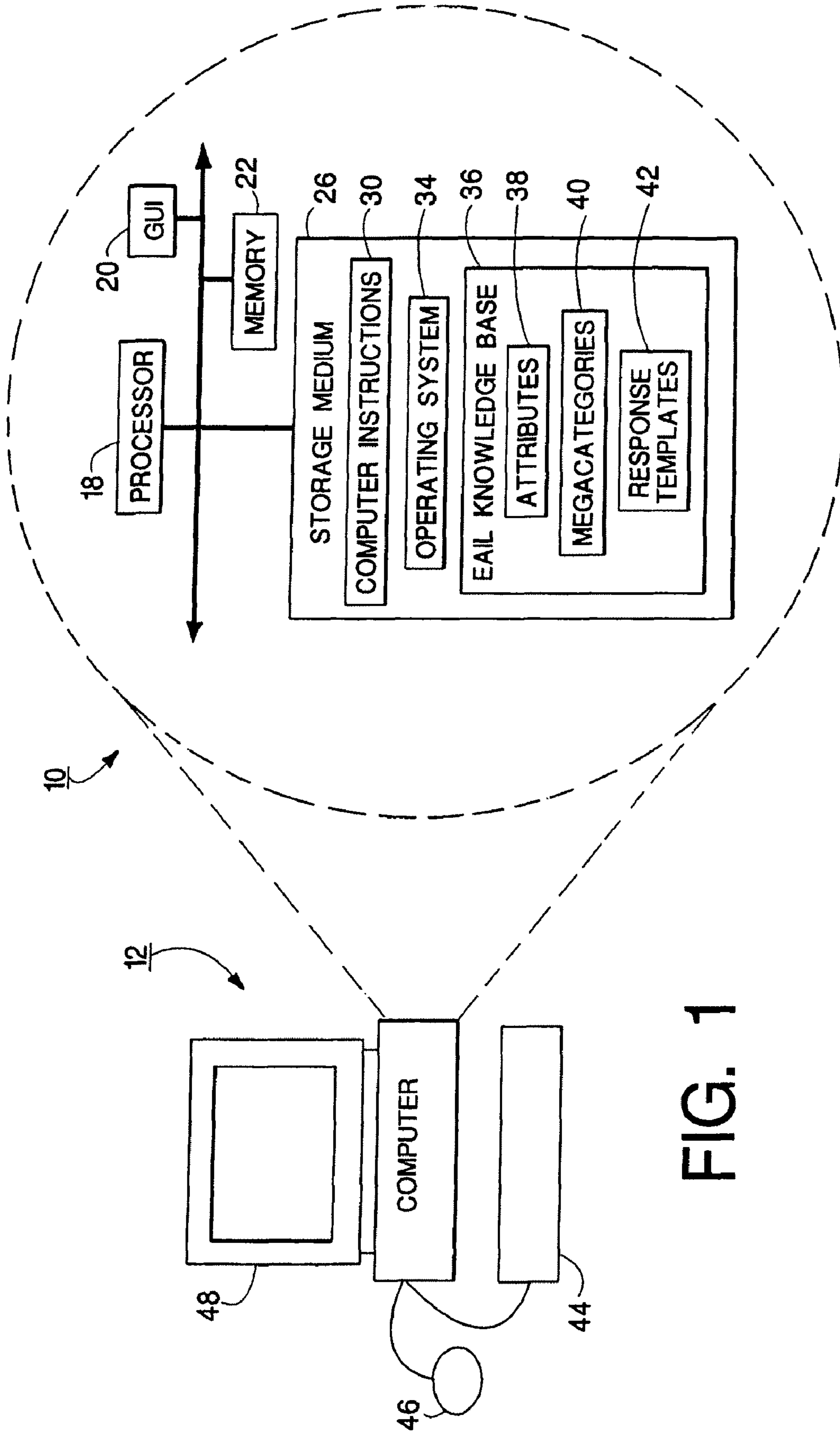


FIG. 1

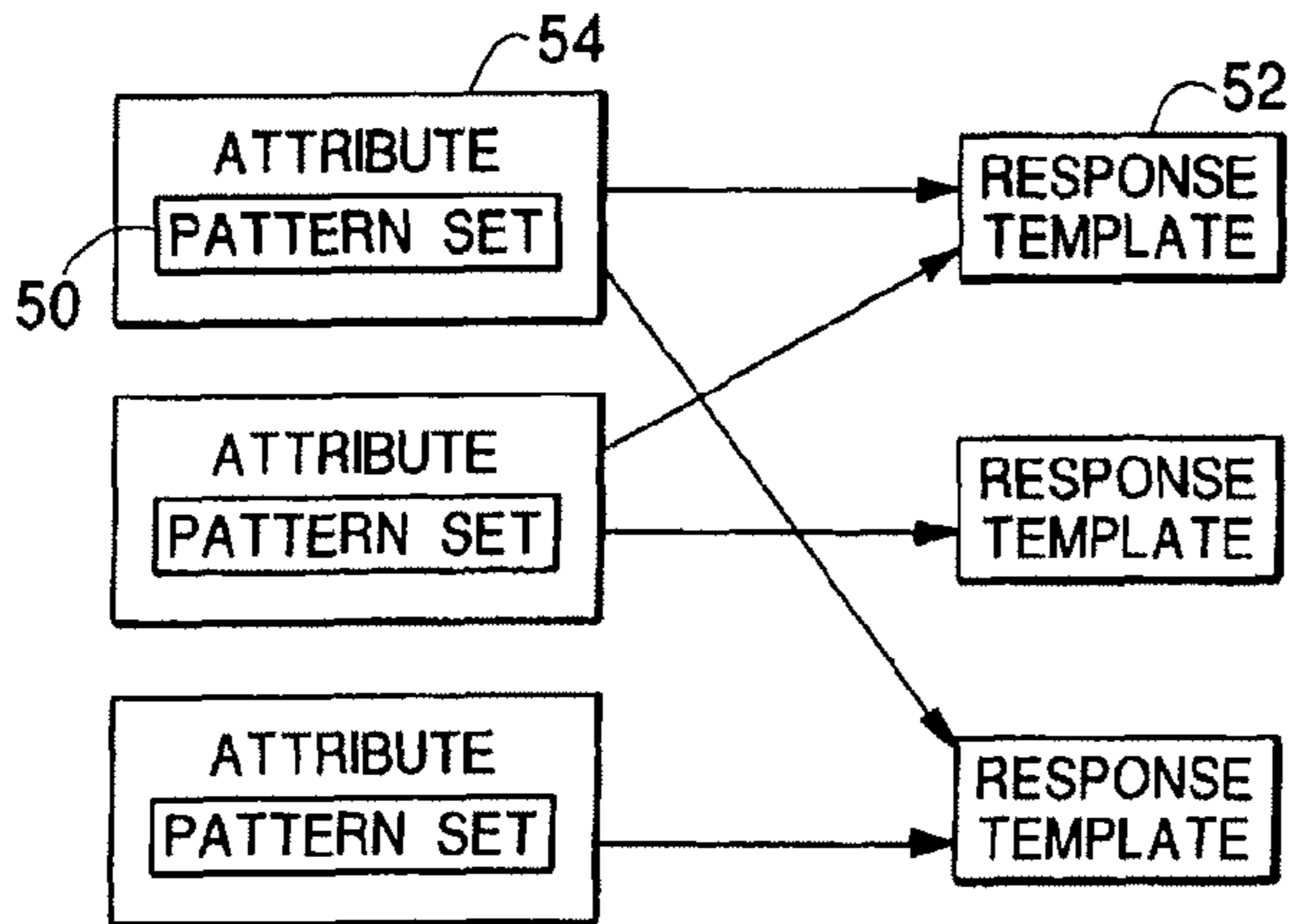


FIG. 2

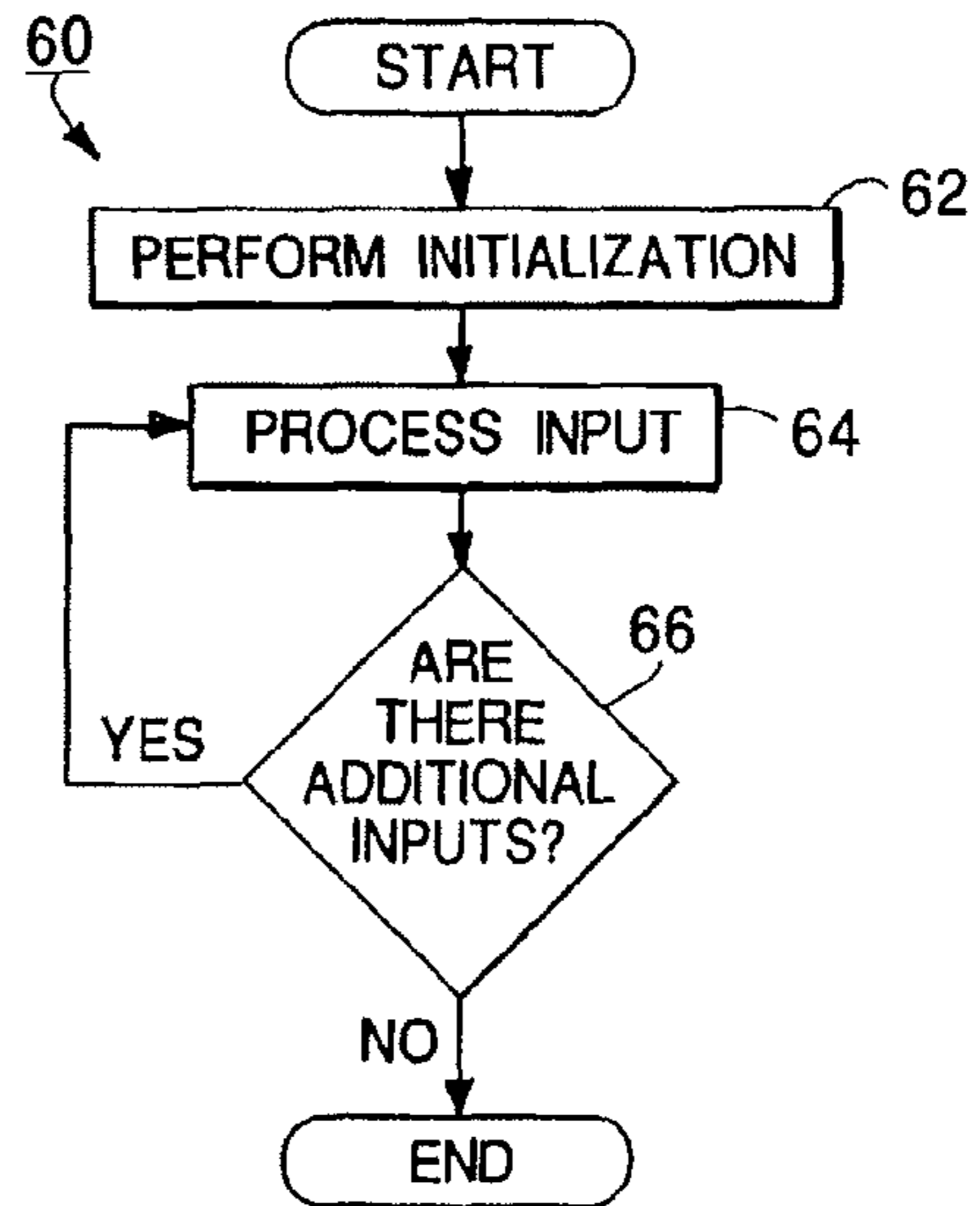


FIG. 3

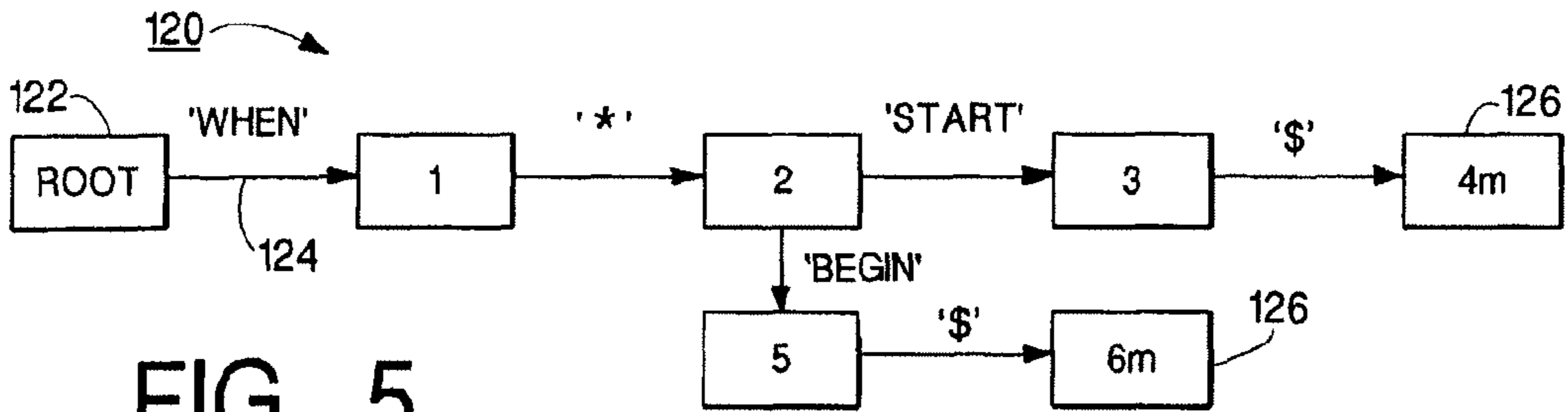


FIG. 5

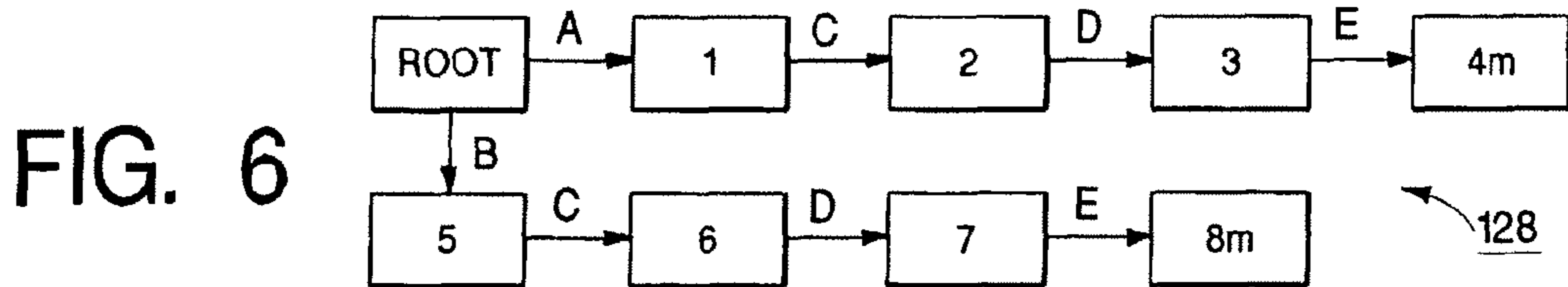


FIG. 6

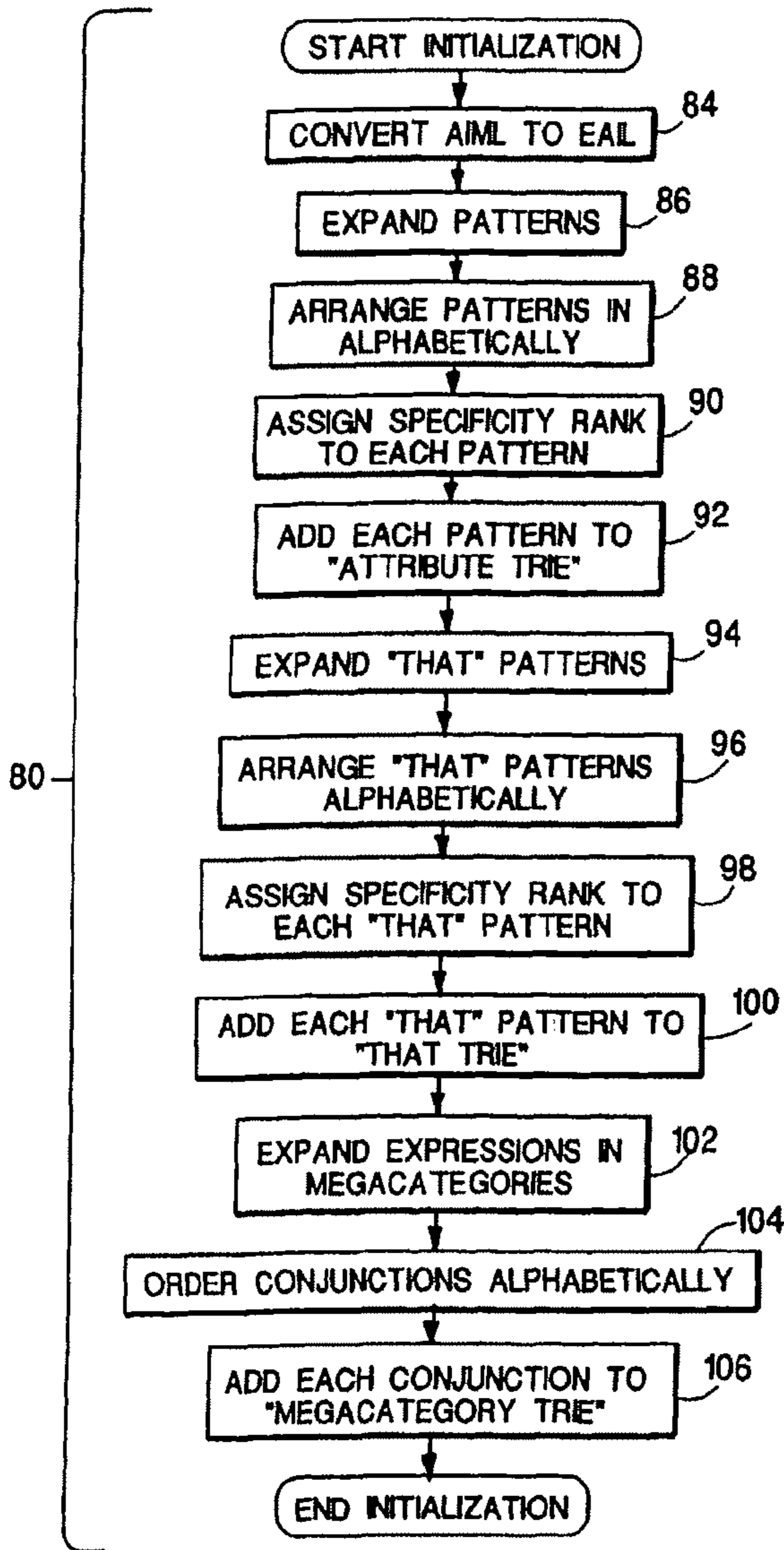


FIG. 4

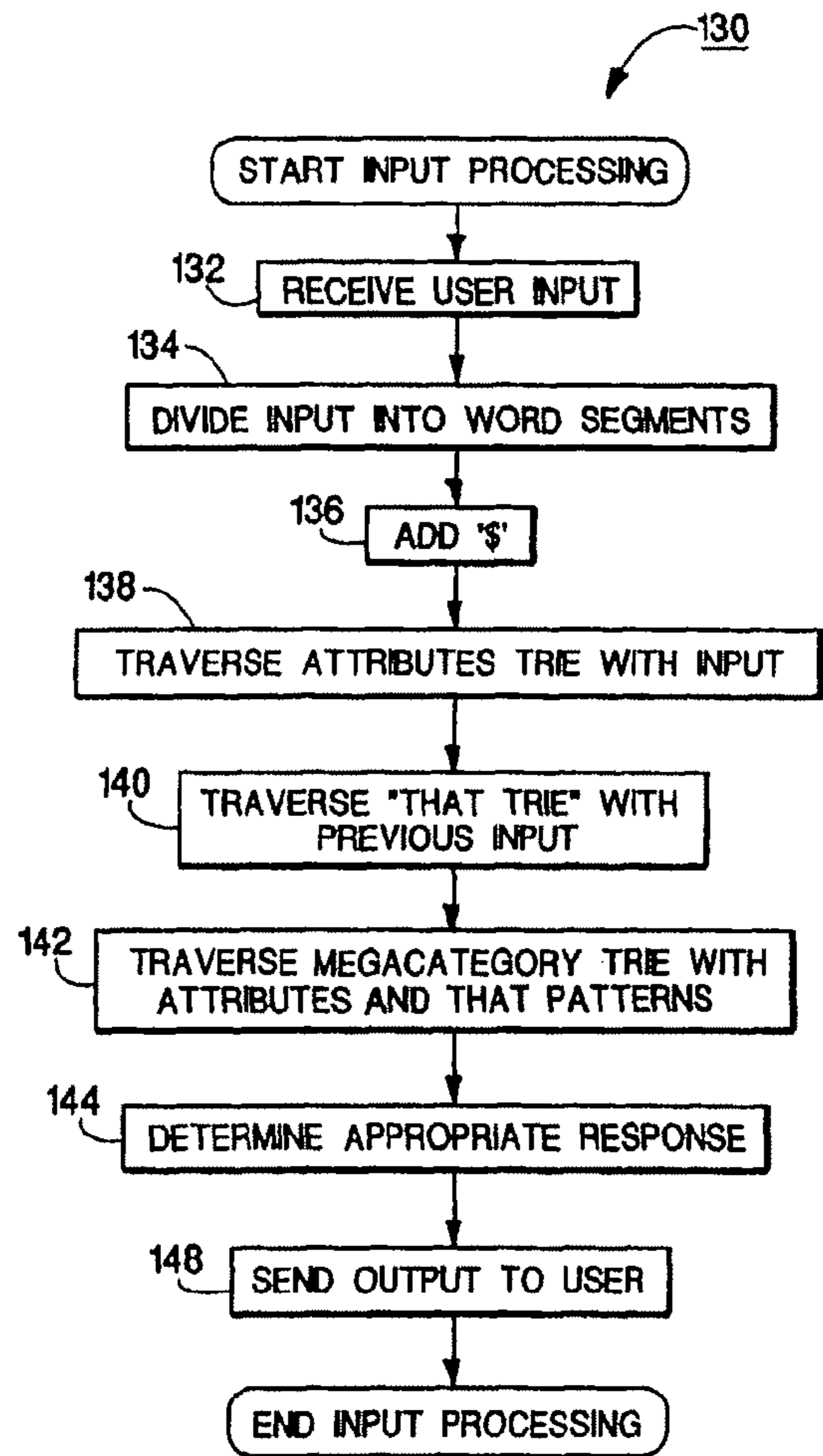


FIG. 7

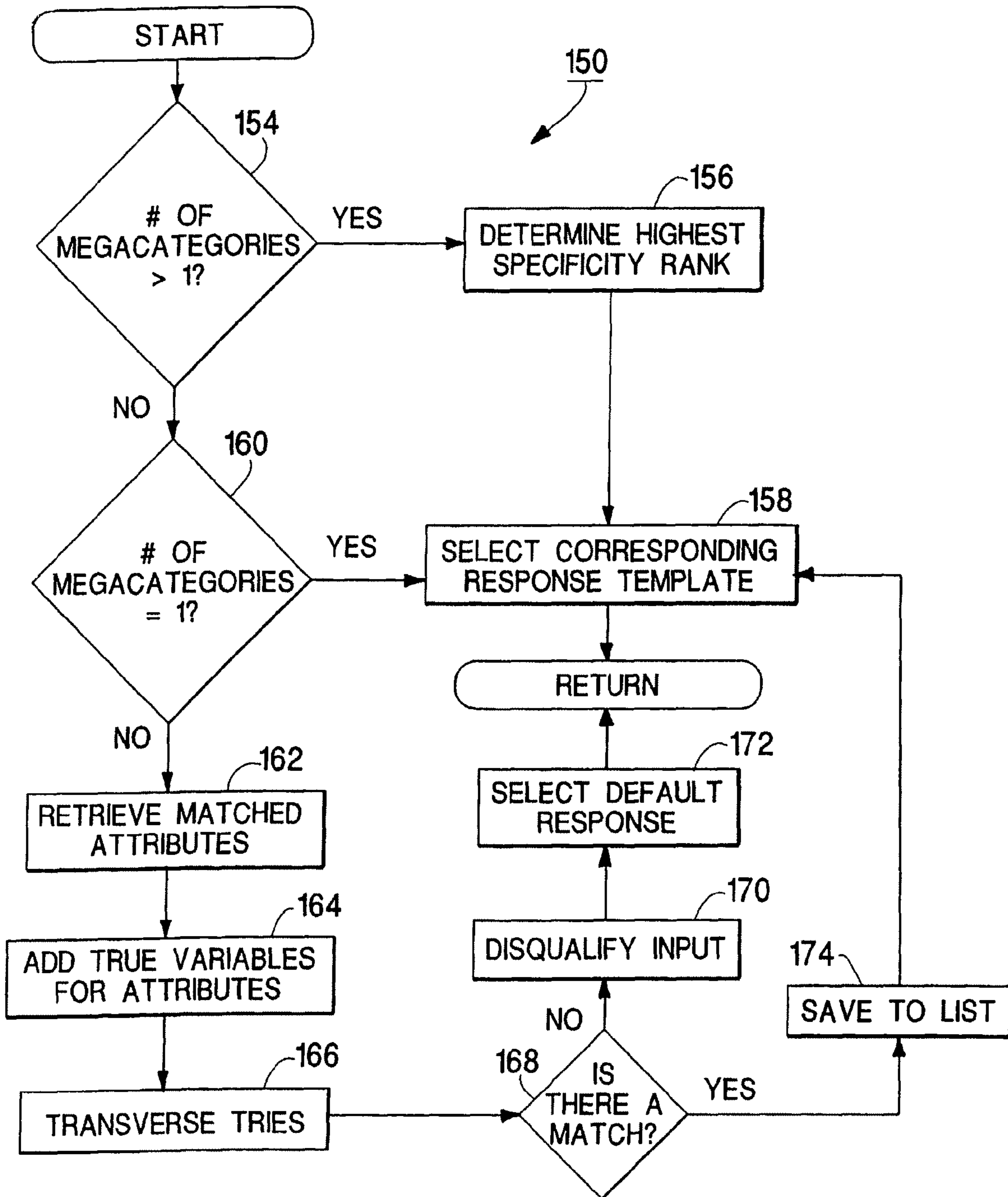


FIG. 8

ENHANCED ARTIFICIAL INTELLIGENCE LANGUAGE

TECHNICAL FIELD

This invention relates to artificial intelligence and more particularly to machines that can use artificial intelligence to converse with humans.

BACKGROUND

Since the development of the computer, human beings have sought to construct computers that can think like humans to the degree that these machines have an “artificial intelligence.” One of the goals is to develop a computer that could think, learn and carry-on intelligent conversations with humans beings.

Some development of artificially intelligent computers focuses on developing computers that can converse. Thus, a key area in developing an artificially intelligent computer is developing a language that allows a computer to process inputs received from humans and to respond with an appropriate and cogent output. One such language is an Artificial Intelligence Mark-up Language (AIML) developed by Dr. Richard S. Wallace.

AIML is interpreted and processed by an engine called A.L.I.C.E. (Artificial Linguistic Internet Computer Entity). The A.L.I.C.E. engine is designed to receive an input from the user and determine the correct response using knowledge encoded in AIML and stored in a AIML knowledge base. For example, when the input is, “What is two plus two?,” the A.L.I.C.E. engine provides the response, “4.” In arriving at the response of “4,” the A.L.I.C.E. engine searches a list of categories within the AIML knowledge base. Each category contains a pattern that is linked to a single response template. The A.L.I.C.E. engine matches the user input against the available patterns in the AIML knowledge base. After finding a match in a pattern, the pattern’s corresponding response template is activated and a series of actions are carried out by the A.L.I.C.E. engine.

SUMMARY

In one aspect, the invention is a method of determining an appropriate response to an input. The method includes linking a plurality of attributes to a plurality of response templates using a plurality of Boolean expressions. Each attribute is associated with a set of patterns. Each pattern within the set of patterns is equivalent. The method also includes determining an appropriate response template from the plurality of response templates based on the input.

In another aspect, the invention is an article that includes a machine-readable medium that stores executable instructions for determining an appropriate response to an input. The instructions cause a machine to link a plurality of attributes to a plurality of response templates using a plurality of Boolean expressions. Each attribute is associated with a set of patterns. Each pattern within the set of patterns is equivalent. The instructions also cause a machine to determine an appropriate response template from the plurality of response templates based on the input.

In a further aspect, the invention is an apparatus that includes a memory that stores executable instructions determining an appropriate response to an input. The apparatus also includes a processor that executes the instructions to link a plurality of attributes to a plurality of response templates using a plurality of Boolean expressions. Each attribute is

associated with a set of patterns. Each pattern within the set of patterns is equivalent. The processor also executes instructions to determine an appropriate response template from the plurality of response templates based on the input.

One or more of the aspects above may have one or more of the following features. Determining an appropriate response template may include performing an initialization. Performing an initialization may include converting an AIML category having one pattern and one response template to an equivalent combination of one attribute and one megacategory, the one megacategory comprising the one response template and storing the equivalent combination.

Performing an initialization may also include expanding the patterns, arranging the patterns in alphabetical order, assigning a specificity rank to each pattern and adding each pattern to an attribute trie. Performing an initialization may further include expanding “that” patterns, arranging “that” patterns alphabetically, assigning a specificity rank to each “that” pattern; and adding each “that” pattern to a “that” trie. Performing an initialization may include expanding megacategories into conjunctions, ordering the conjunctions alphabetically and adding each megacategory to a megacategory trie.

The linking of the plurality of attributes to the plurality of response templates may include forming a megacategory that links a combination of attributes to one of the plurality of response templates using one of the plurality of Boolean expressions. The linking of the attributes to the plurality of response templates may include forming a megacategory that links one attribute to one of the plurality of response templates. The megacategory may include a “that” tag that contains a pattern that is syntactically equivalent to patterns within attributes.

The combination of attributes using one of the plurality of Boolean expressions may include a name for each attribute in the combination and an operator. The operator may be an AND operator. The megacategory may include a hierarchy of levels in matching the megacategory to the input.

Each aspect may include grouping a plurality of megacategories into a topic having a topic name and adding a topic variable to one of the plurality of response templates.

Each aspect may also include comparing a megacategory to the input when the topic name matches the topic variable. “Being equivalent” may include being semantically equivalent.

Each aspect may further include storing the plurality of attributes and the plurality of response templates. Each aspect may further include executing the appropriate response template.

Determining an appropriate response template may include dividing the input into word segments, traversing the attribute trie with the input to determine matching attributes, traversing the “that” trie to determine matching “that” patterns from a previous output and traversing the megacategory trie with matching attributes patterns and matching “that” patterns to determine matching megacategories.

Traversing the attribute trie may include assigning a variable to each pattern and setting the variable to true for each pattern that matches a portion of the input.

If one megacategory matches, a response template may be selected corresponding to the one megacategory. If more than one megacategory matches, a response template may be selected corresponding to a megacategory having the highest summed specificity rank from the set of patterns. If more than one megacategory has the highest summed specificity rank, a megacategory may be selected having a pattern having the single highest specificity rank. If no megacategories match,

matched attributes may be added from a last user input to the current matched attributes and determining a matching megacategory.

One or more of the aspects above have one or more of the following advantages. An enhanced artificial intelligence language engine can accept knowledge base data from existing languages such as AIML code. The enhanced artificial intelligence language allows for the implicit recognition of contextual information. The enhanced artificial intelligence language extends the pattern matching capabilities of AIML, and allows knowledge base developers to reuse portions of a knowledge base for generating other knowledge bases. Since the enhanced artificial intelligence language is a modular language, portions of the knowledge base can be modified without modifying the entire knowledge base.

DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system for processing an enhanced artificial intelligence language.

FIG. 2 is a diagram of the relationship between attributes and response templates in the enhanced artificial intelligence language (EAIL).

FIG. 3 is a process for generating a response.

FIG. 4 is an initialization process.

FIG. 5 is an example of an attribute trie used in determining matching patterns.

FIG. 6 is an example of a megacategory trie used in selecting a response template.

FIG. 7 is a process for processing an input.

FIG. 8 is a process for selecting a response template.

DESCRIPTION

Referring to FIG. 1, an enhanced artificial intelligence language (EAIL) engine 10, within a system 12 (e.g., a computer), is used to process user inputs and to generate appropriate responses using a process 60 (FIG. 3). EAIL engine 10 includes a processor 18, a graphical user interface (GUI) 20, a memory 22, and a storage medium 26 (e.g., a hard disk). Storage medium 26 stores a set of computer instructions 30, an operating system 34 and an EAIL knowledge base 36. EAIL knowledge base 36 includes attributes 38, megacategories 40 and response templates 42. Each of the components within knowledge base 36 is further discussed below. A knowledge base forms a unique artificial personality.

EAIL engine 10 receives a user's inputs. The user inputs a question or statement using an input/output (I/O) interface, such as a keyboard 44 and/or a mouse 46, using a GUI 20. GUI 20 renders a computer image of a cyber personality on display 48 to give a user a semblance of an intelligent entity. For example, the cyber personality could be a human 3-dimensional face. The set of computer instructions 30 is executed by processor 18 out of memory 22 to process the inputs using knowledge base 36 to determine the matching megacategory in order to generate an appropriate response.

Referring to FIG. 2, in EAIL engine 10, a pattern is not necessarily tied to a single response template 52. Instead, EAIL engine 10 includes attributes 54. An attribute 54 has a set of patterns 50 that are semantically (though not syntactically) equivalent within an attribute 54. Each attribute is the association of a pattern of letters with their meaning. For example, a "hello" attribute contains a sequence of letters that have been defined to mean hello (e.g., hi, howdy, bonjour, etc.).

In EAIL engine 10, categories link attributes 54 to response templates 52 using Boolean expressions. In these expres-

sions, the atomic propositions are attributes 38. Categories that use combinations of attributes, instead of a single pattern, are called megacategories 40.

Since multiple attributes can be used in combination, the structure of EAIL knowledge base 36 includes attributes, and response templates and a set of rules for mapping attributes to response templates. By decoupling patterns from responses, and grouping them into semantically equivalent attributes, it is easier to re-use patterns for knowledge bases with different artificial personalities.

EAIL 10 searches attributes to match the input to the appropriate attribute. Even when no megacategories match (i.e., no Boolean expression evaluated is true), EAIL engine 10 looks at the attributes that matched the previous input. These attributes can be used in conjunction with the attributes that matched the current input. Thus, EAIL engine 10 can determine context without the knowledge base coder having to explicitly set variables beforehand.

EAIL engine 10 has a modular approach, where a set of attributes can be generated and distributed without having any affect on any one artificial personality. Since pattern matching in EAIL engine 10 is done independently of megacategories, multiple wildcards and more general regular expression operators can be added while still employing most of AIML's category selection algorithm.

I. EAIL Knowledge Base Overview

A. EAIL Knowledge Base Syntax

EAIL uses an XML-based format to encode knowledge. EAIL files have an ".eail" extension and have a root element named <eail>. A partial document type definition (DTD) for EAIL appears below. The definition of the contents of the <template> tag and the <category> tag, not shown, are identical to their AIML 1.0 analogs except for the <star> tag.

```
<!ELEMENT eail (attribute*, category*, megacategory*,
topic*)>
<!ELEMENT attribute (pattern+)>
<!ELEMENT megacategory (that, expr, template)>
<!ELEMENT topic (category*, megacategory*)>
<!ELEMENT pattern (#PCDATA)>
<!ELEMENT that (#PCDATA)>
<!ELEMENT expr (#PCDATA)>
<!ATTLIST attribute
  name CDATA #REQUIRED>
<!ATTLIST megacategory
  level CDATA #IMPLIED>
<!ATTLIST pattern
  sindex CDATA #IMPLIED>
```

B. Attributes

Each attribute 38 in EAIL knowledge base 36 has a name and a set of one or more patterns 40. The name of an attribute may be any combination of letters, numbers and an underscore. One or more of the following symbols may appear in a pattern. An "*" symbol, (asterisk), denotes a wildcard and matches one or more words in the input. An "_" symbol, (underscore), is also a wildcard, and provides the same function as the asterisk except the underscore lexicographically precedes all alphanumeric characters, whereas the asterisk follows alphanumeric characters. A "?" symbol, (question mark), immediately after a word makes the word optional. A "{ . . . | . . . }" symbol represents series of words that appear between "{" and "}" and are separated by "|" to denote a selection of one of the words in the series. A "%" symbol, a percent symbol, can appear immediately before or after a word to show that the word can be a sub-string of a word in the

5

input. For example, “% comfort %” would match “comfort,” “comfortable”, and “uncomfortable”.

Each pattern may also have a “sindex” attribute. The “sindex” attribute is used to determine which asterisk to use when the <star> tag appears in a response template. Since there are multiple wildcards in a pattern, ambiguity can arise unless it is clear which wildcard is meant to be used. In the absence of this attribute, the value 1 is assumed (1 is the first wildcard, not 0).

C. Megacategories

Each megacategory **40** links a combination of attributes **38** to a single response template **42**. The combination is represented by an expression having attributes’ names, the AND operator (&&), and the OR operator (||). Parentheses are allowed, but the unary NOT (!) is not allowed. These expressions will be converted to disjunctive normal form by EAIL engine **10** for efficient evaluation.

Each megacategory may optionally have a “that” tag, which is semantically equivalent to the “that” tag in AIML categories. The “that” tag contains a pattern that matches against a previous artificial personality output that allows a programmer to put in different responses to the same input based on what the artificial personality just said. For example, a programmer wants to program a knock-knock joke. The artificial personality will start by saying “knock, knock.” Then the user will say “Who’s there?” When programming a response to the pattern “Who’s there?,” the programmer would put a “that” tag in with the pattern “knock, knock.” If the programmer leaves out the “that” tag, the artificial personality will try to tell knock-knock jokes any time the user says “Who’s there?,” which is not the desired behavior. For instance, if the artificial personality says “I’m going to visit someone in Florida,” and the user says “Who’s there?,” it would be inappropriate for the artificial personality to tell a knock-knock joke.

A programmer designing knowledge base **36** may also optionally assign a level to each megacategory. Giving megacategories different levels imposes an explicit hierarchy on the order in which megacategories are evaluated by EAIL engine **10**. As a default, all megacategories are set at level one. All megacategories of level one are considered before megacategories of level **2**, and all megacategories of level **2** are considered before megacategories of level **3**, etc.

The response template for a megacategory is identical to the response template for an AIML category, except in the treatment of the <star> tag. Since there are possibly multiple attributes working together in a megacategory, the <star> tag references the name of the attribute that contains the wildcard it is referencing. Within the attribute, the pattern tag contains information about which specific asterisk to use (i.e., “sindex” attribute). When multiple patterns within an attribute match the input, and they all have a “sindex” attribute, the pattern with the highest specificity rank has precedent.

D. Categories

Categories have the same knowledge that was originally intended for use by an AIML engine. In order to be used in an EAIL knowledge base, each category is converted to an equivalent combination of an attribute and a megacategory. This conversion is carried out by EAIL engine **10**, and is transparent to the knowledge base code developer.

E. Topics

Megacategories and categories can be grouped into topics, which are identical to topics in AIML. A topic is identified by its name. A special “topic” variable may be set in a response template. Megacategories and categories that appear within a <topic> tag will only match when the topic variable matches the name of the topic.

6

II. EAIL Engine Processes

A. Category Conversion

Referring to FIGS. **3** and **4**, process **60** performs (**62**) an initialization using a process **80**. Process **80** converts (**84**) all of the AIML categories in EAIL knowledge base **36** that were intended for the A.L.I.C.E. AIML interpreter, to EAIL megacategories **40**, before any patterns are matched or any response is generated. Each category includes a pattern and a response template. A new attribute, with an automatically generated name, is generated with the one pattern. A new megacategory is also generated having the same response template as the original category (with <star>expressions converted to be compatible with EAIL). The expression is the name of the newly generated attribute. For example, given:

```
<category>
<that>WHAT CAN I HELP YOU WITH</that>
<pattern>PLEASE *</pattern>
<template><srai><star></srai></template>
</category>
```

is converted into:

```
<attribute name="abcd">
<pattern sindex="1">PLEASE *</pattern>
</attribute>
<megacategory>
<that>WHAT CAN I HELP YOU WITH</that>
<expr>abcd</expr>
<template><srai><star>
attribute="abcd"/</srai></template>
</megacategory>
```

where “abcd” is an automatically generated name for the attribute.

Matching the patterns to the appropriate response is data in which there is one node for every common prefix. The strings are stored in extra leaf nodes.

An attribute trie structure is generated during EAIL engine initialization. The trie has all of the patterns from within all the attributes in EAIL knowledge base **36**. Before the patterns are inserted into the attribute trie, process **80** expands (**86**) the patterns into patterns that do not contain any ‘{ . . . | . . . }’ or ‘?’ symbols. These symbols are removed by generating a set of patterns that as a whole perform the same function before removing any symbols. For instance, the pattern “A B? C” is expanded into two patterns: “A C” and “A B C”. In another instance, the pattern “A {B|C}” is expanded to “A B” and “A C”. In addition to the expansion, a special ‘\$’ symbol is appended to each pattern, to represent the end of the input.

Process **80** arranges (**88**) all of the patterns in the knowledge base **36** in alphabetical order. Process **80** assigns (**90**) a specificity rank that represents its position within the alphabetical list. The specificity rank is used to determine a pattern’s “specificity” in relation to other patterns when choosing between multiple matching megacategories as described below (see FIG. **8**). Process **80** adds (**92**) each pattern to a trie to form an attribute trie.

The following is one example to generate an attribute trie:

```
for each pattern in set
  current_node := root
  for each word w in pattern
```


7

-continued

```

    if (branch exists from current__node on w)
        current__node := node attached to branch
        continue
    else
        generate new__node
        generate a branch from current__node to
new__node on w
        current__node := new__node
    end if
end for
set current__node to be a "match" node
end for

```

FIG. 5 shows a finished attribute trie **120** for the patterns "WHEN*START \$" and "WHEN*BEGIN \$". Each box above represents a node **122**, and each arrow represents a branch **124**. Nodes with a 'm' in their name (nodes **4** and **6**) are match nodes **126**. As will be described further below, reaching these matched nodes, during a traversal of the attribute trie, indicates that a pattern has been matched.

During initialization a "that" trie is generated similarly to the attribute trie. Process **80** expands (**94**) the "that" patterns. Process **80** arranges (**96**) the "that" patterns alphabetically. Process **80** assigns (**98**) a specificity rank to each "that" pattern. Process **80** adds (**100**) each pattern to a trie to form a "that" trie.

During initialization a megacategory trie is also generated. Process **80** expands (**102**) expressions in the megacategories into the disjunctive normal form. Process **80** orders (**104**) the conjunctions alphabetically. Process **80** adds (**106**) each conjunction to a trie to form a "megacategory" trie. A separate trie for each "level" of megacategories in the EAIL knowledge base is generated.

For example of generating a megacategory trie, consider a megacategory that has the expression "C && (A|B)", a "that" tag, and a topic. A variable is generated by EAIL engine **10** for both the "that" tag and the topic. Suppose the variable for the "that" tag is "D" and the variable for the topic is "E". EAIL engine **10** "AND's" the original expression with both variables. The expression becomes "((C && (A|B)) && D) && E".

This compound expression is expanded to the DNF. In this example, the resulting expression would be "(C && A && D && E)|(C && B && D && E)". Each conjunction is ordered alphabetically by variable name, so each conjunction now becomes "(A && C && D && E)|(B && C && D && E)". The conjunctions (two in all for this example) are added to a trie (not shown) constructed similarly to a trie **128** in FIG. **6**. Each matching node maintains a link to the megacategory with which it is associated.

B. Processing an Input

Referring to FIGS. **3** and **7**, process **60** processes (**54**) an input using a process **130** by traversing the tries (e.g., attribute trie, that trie, and megacategory trie) generated during initialization (e.g., process **80**). Process **130** receives (**132**) an input.

To traverse the attribute trie, process **130** divides (**134**) the input up into words. Punctuation is removed and spaces between words in the input are treated as a word delimiter. Process **110** appends (**136**) the words with a '\$' symbol.

8

Process **110** traverses (**138**) the attribute trie with the input to obtain a list of matching attributes. One example of traversing the attribute trie uses the following:

```

procedure findPatterns(input)
    findPatterns(root, input, 0);
end procedure
procedure findPatterns(current__node, input, pos+2)
    if (current__node is match node)
        pattern found
    end if
    cur__word := word at position pos in input
    for each branch from current__node on cur__word
        new__node = node attached at branch
        findPatterns(new__node, input, pos+1)
    end for
    if (branch exists from current node on "*" or "_")
        new__node = node attached at branch
        resolveWildCard(new__node, input, pos)
    end if
end procedure
procedure resolveWildCard(current__node, input, pos)
    for each word w in input starting at position pos
        if (branch exists from current__node on w)
            new__node = node attached at branch
            new__pos := position of w in input
            findPatterns(new__node, input, new__pos)
        end if
    end for
end procedure

```

Process **130** traverses (**140**) the "that" trie with the previous output of EAIL engine **10** to obtain a list of matching "that" patterns.

Process **130** traverses (**142**) the megacategory trie with matched attributes and matched "that" patterns (and with topic variables if applicable) to obtain a list of matching megacategories. In traversing trie **128**, the following is used to find all matching expressions:

```

procedure findExpressions(vars__list)
    for pos1 := 0 to length of vars__list
        current__node := root
        for pos2 := pos1 to length of vars__list
            var := variable at pos2 in vars__list
            if (current__node is match)
                matching expression found
            end if
            if (branch exists from current__node on var)
                current__node := node attached at branch
            else
                break
            end if
        end for
    end for
end procedure

```

Each generated level of megacategory trie is traversed. The megacategory tries corresponding to level one are traversed first. If no matching megacategories are found, then level **2** megacategory tries are traversed, and so on until there are no more megacategory tries to traverse. When this procedure concludes, a set of megacategories remains that are matched to the current input.

C. Determining an Appropriate Response Template

Referring to FIGS. **7** and **8**, process **80** determines (**144**) an appropriate response using a process **150**. Process **150** determines (**154**) if more than one megacategory matches. If more than one megacategory matches, process **150** determines (**156**) the highest "specificity rank" from each of the matching megacategories. The specificity rank is determined for each

of the patterns during EAIL engine initialization in process **80** (see FIG. 4). The specificity rank of a megacategory is equal to the sum of the specificity ranks of the patterns that contributed to the matching of the megacategory. In the case of a tie, the megacategory with the highest single ranking for a specific pattern is used. Process **150** selects (**158**) the corresponding response template.

Process **150** determines (**160**) if only one megacategory has been matched. If only one megacategory has been matched process **150** selects (**158**) the corresponding response template to the matched megacategory.

Otherwise, no megacategories are matched. EAIL engine **10** stores a list of matched attributes for each user input. Process **150** retrieves (**162**) the matched attributes from the previous user input. Process **150** adds (**164**) the matched attributes from the previous input to a current list of matched attributes. Only matched attributes are propagated forward (i.e., topic variables and “that” tag variables do not propagate forward from previous inputs). After previous matched attributes are added to the current list, process **150** traverses (**166**) the tries for each level. Thus, EAIL engine **10** is “implicitly” determining context.

For example, consider a knowledge base with the following data:

```

<eail>
  <attribute name="head">
    <pattern>* HEAD *</pattern>
  </attribute>
  <attribute name="pain">
    <pattern>* HURT% *</pattern>
  </attribute>
  <attribute name="duration">
    <pattern>WHEN * {START | BEGIN}</pattern>
    <pattern>HOW LONG *</pattern>
  </attribute>
  <megacategory>
    <expr>head</expr>
    <template>My head hurts.</template>
  </megacategory>
  <megacategory>
    <expr>head && pain && duration</expr>
    <template>My head has been hurting for about a
day</template>
  </megacategory>
</eail>

```

If the user types in, “Tell me about your head,” only the “head” variable will be set to true. The one megacategory with “head” as its expression will match, and EAIL engine **10** responds “My head hurts.” If the user types in, “How long has it been hurting,” the “pain” and “duration” variables will artificial both be set to true. However, there are no megacategories that match just those two variables. EAIL engine **10** adds “head,” from the last input, to the current list of matched attributes. The megacategory with “head && pain && duration” now matches, and EAIL engine **10** responds “My head has been hurting for about a day.” In this way, EAIL engine **10** determined that the user was talking about the head without the knowledge base code developer having to explicitly set a topic or a “that” tag.

Process **150** determines (**168**) if there are any megacategories that match. If there are still no matching megacategories, process **150** disqualifies (**170**) the input. This ensures that EAIL engine **10** does not give the same response twice in a row when no matched attributes occur for a given input. Process **150** selects (**172**) a default response so that when executed responds to the user with a “request for clarification” response.

If a pattern matches on the second traversal, process **150** saves (**174**) the attributes that match permanently to a list of “current” matched attributes. Thus, if the user asked another question about the “head”, as in the example described above, “head” would be a part of the previous list of matched attributes.

Referring back to FIGS. 3 and 4, process **80** sends (**144**) the appropriate response. Process **60** determines (**66**) if there are additional inputs. If there are additional inputs, process **60** processes (**64**) the input.

Other Processing Embodiments

Processes **60**, **80**, **130** and **150** are not limited to use with the hardware and software of FIG. 1; it may find applicability in any computing or processing environment and with any type of machine that is capable of running a computer program. Processes **60**, **80**, **130** and **150** may be implemented in hardware, software, firmware or any combination of the three. For example, processes **60**, **80**, **130** and **150** may be implemented in a circuit that includes one or a combination of a processor, a memory, programmable logic and logic gates. Processes **60**, **80**, **130** and **150** may be implemented in computer programs executed on programmable computers/machines that each includes a processor, a storage medium or other article of manufacture that is readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices. Program code may be applied to data entered using an input device to perform processes **60**, **80**, **130** and **150** and to generate output information.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language. The language may be a compiled or an interpreted language. Each computer program may be stored on a storage medium or device (e.g., CD-ROM, hard disk, or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform processes **60**, **80**, **130** and **150**. Processes **60**, **80**, **130** and **150** may also be implemented as a machine-readable storage medium, configured with a computer program, where upon execution, instructions in the computer program cause the computer to operate in accordance with processes **60**, **80**, **130** and **150**.

The process is not limited to the specific embodiments described herein. For example, GUI **20** does not need to display a three-dimensional person to depict the cyber personality. The dialogue between EAIL **10** and the user can be in the form of text messages on a screen. The process is not limited to the specific processing order of FIGS. 3, 4, 7 and 8. Rather, the blocks of FIGS. 3, 4, 7 and 8 may be re-ordered, as necessary, to achieve the results set forth above.

Other embodiments not described herein are also within the scope of the following claims.

What is claimed is:

1. A computer implemented method of determining an appropriate response to an input, comprising:
 - linking by a computer system a plurality of attributes to a plurality of response templates using a plurality of Boolean expressions, each attribute associated with a set of patterns, with each pattern within the set of patterns being equivalent;

11

receiving by the computer the input; and
determining by the computer system an appropriate
response template for the received input, the appropriate
response template selected from the plurality of
response templates based on the input, by: 5
determining by the computer system matches of the input
to patterns;
ranking by the computer system the patterns; and
selecting by the computer the corresponding response tem-
plate based on the ranked patterns. 10
2. The method of claim **1**, wherein determining comprises:
performing an initialization by the computer.
3. The method of claim **2** wherein linking the plurality of
attributes to the plurality of response templates comprises: 15
forming by the computer system a megacategory, with the
megacategory linking a combination of attributes to one
of the plurality of response templates using one of the
plurality of Boolean expressions.
4. The method claim **2** wherein linking the plurality of 20
attributes to the plurality of response templates comprises:
forming by the computer system a megacategory, with the
megacategory linking one attribute to one of the plural-
ity of response templates.
5. The method of claim **3** wherein the combination of 25
attributes using one of the plurality of Boolean expressions
comprises:
a name for each attribute in the combination; and
an operator Boolean expression.
6. The method of claim **5** wherein the operator is an AND 30
operator.
7. The method of claim **3** wherein the megacategory com-
prises:
a hierarchy of levels in matching the megacategory to the
input. 35
8. The method of claim **3**, further comprising:
grouping a plurality of megacategories into a topic having
a topic name; and
adding a topic variable to one of the plurality of response
templates. 40
9. The method of claim **8**, further comprising:
comparing by the computer a megacategory to the input
when the topic name matches the topic variable.
10. The method of claim **1** wherein being equivalent com-
prises being semantically equivalent. 45
11. The method of claim **1**, further comprising:
storing in computer storage the plurality of attributes and
the plurality of response templates.
12. The method of claim **3**, wherein performing an initial-
ization comprises: 50
converting an AIML category having one pattern and one
response template to an equivalent combination of one
attribute and one megacategory, the one megacategory
comprising the one response template; and
storing the equivalent combination. 55
13. The method of claim **3** wherein performing an initial-
ization comprises:
expanding the patterns;
arranging the patterns in alphabetical order;
assigning a specificity rank to each pattern; and
adding each pattern to an attribute trie. 60
14. The method of claim **13** wherein performing an initial-
ization comprises:
expanding “that” patterns;
arranging “that” patterns alphabetically;
assigning a specificity rank to each “that” pattern; and
adding each “that” pattern to a “that” trie. 65

12

15. The method of claim **14** wherein performing an initial-
ization comprises:
expanding megacategories into conjunctions;
ordering the conjunctions alphabetically; and
adding each megacategory to a megacategory trie. 5
16. The method of claim **15**, wherein determining an
appropriate response template comprises:
dividing the input into word segments;
traversing the attribute trie with the input to determine
matching attributes; 10
traversing the “that” trie to determine matching “that” pat-
terns from a previous output;
traversing the megacategory trie with matching attributes
patterns and matching “that” patterns to determine
matching megacategories.
17. The method of claim **16** wherein traversing the attribute
trie comprises:
assigning a variable to each pattern; and
setting the variable to true for each pattern that matches a
portion of the input.
18. The method of claim **16**, further comprising:
if one megacategory matches, selecting a response tem-
plate corresponding to the one megacategory.
19. The method of claim **16**, further comprising:
if more than one megacategory matches, selecting a
response template corresponding to a megacategory
having the highest summed specificity rank from the set
of patterns.
20. The method of claim **19**, further comprising:
if more than one megacategory has the highest summed
specificity rank, selecting a megacategory having a pat-
tern having the single highest specificity rank.
21. The method of claim **16**, further comprising:
if no megacategories match, adding matched attributes
from a last user input to the current matched attributes;
and
determining a matching megacategory.
22. The method of claim **1**, further comprising:
executing the appropriate response template.
23. The method of claim **3**, wherein the megacategory
comprises a “that” tag containing a pattern that is syntacti-
cally equivalent to patterns within attributes.
24. An article comprising a machine readable storage
device that stores executable instructions for determining an
appropriate response to an input, the instructions causing a
processor to:
link a plurality of attributes to a plurality of response tem-
plates using a plurality of Boolean expressions, each
attribute associated with a set of patterns, each pattern
within the set of patterns being equivalent;
receive the input; and
determine an appropriate response template for the
received input, the appropriate response template
selected from the plurality of response templates based
on the input, by instructions to:
determine matches of the input to patterns;
rank the patterns; and
select the corresponding response template based on the
ranked patterns .
25. The article of claim **24**, wherein instructions causing a
machine to determine comprises:
performing an initialization.
26. The article of claim **25** wherein instructions causing a
machine to link the plurality of attributes to the plurality of
response templates comprises instructions to:

13

form a megacategory, with the megacategory linking a combination of attributes to one of the plurality of response templates using one of the plurality of Boolean expressions.

27. The article of claim 24 wherein being equivalent comprises being semantically equivalent.

28. The article of claim 24, further comprising instructions causing a machine to:

store the plurality of attributes and the plurality of response templates.

29. The article of claim 26, wherein instructions to perform an initialization comprise instructions to:

convert an AIML category having one pattern and one response template to an equivalent combination of one attribute and one megacategory, the one megacategory comprising the one response template; and

store the equivalent combination.

30. The article of claim 26 wherein instructions to perform an initialization comprises instructions to:

expand the patterns;
arrange the patterns in alphabetical order;
assign a specificity rank to each pattern; and
add each pattern to an attribute trie.

31. The article of claim 30 wherein instructions to perform an initialization comprises instructions to:

expand "that" patterns;
arrange "that" patterns alphabetically;
assign a specificity rank to each "that" pattern; and
add each "that" pattern to a "that" trie.

32. The article of claim 31 wherein instructions to perform an initialization comprises instructions to:

expand megacategories into conjunctions;
order the conjunctions alphabetically; and
add each megacategory to a megacategory trie.

33. The article of claim 32, wherein instructions causing a machine to determine an appropriate response template comprises instructions to:

divide the input into word segments;
traverse the attribute trie with the input to determine matching attributes;
traverse the "that" trie to determine matching "that" patterns from a previous output; and
traverse the megacategory trie with matching attributes patterns and matching "that" patterns to determine matching megacategories.

34. The article of claim 33 wherein instructions to traverse the attribute trie comprises instructions to:

assign a variable to each pattern; and
set the variable to true for each pattern that matches a portion of the input.

35. The article of claim 33, further comprising instructions causing a machine to:

determine that one megacategory matches, and when there is a determined match,
select a response template corresponding to the one megacategory.

36. The article of claim 33, further comprising instructions causing a machine to:

determine that more than one megacategory matches, and when there are determined matches,
select a response template corresponding to a megacategory having the highest summed specificity rank from the set of patterns.

37. The article of claim 36, further comprising instructions causing a machine to:

determine that more than one megacategory has the highest summed specificity rank,

14

select a megacategory having a pattern having the single highest specificity rank.

38. The article of claim 33, further comprising instructions causing a machine to:

determine that no megacategories match,
add matched attributes from a last user input to the current matched attributes; and
determine a matching megacategory.

39. The article of claim 24, further comprising instructions causing a machine to:

execute the appropriate response template.

40. An apparatus comprising:

a memory that stores executable instructions for determining an appropriate response to an input; and
a processor that executes the instructions to:

link a plurality of attributes to a plurality of response templates using a plurality of Boolean expressions, each attribute associated with a set of patterns, with each pattern within the set of patterns being equivalent;

receive the input; and

determine an appropriate response template for the received input, the appropriate response template selected from the plurality of response templates based on the input, by instructions to:

determine matches of the input to patterns;
rank the patterns; and

select the corresponding response template based on the ranked patterns.

41. The apparatus of claim 40, wherein instructions to determine comprise instructions to:

perform an initialization.

42. The apparatus of claim 41 wherein instructions to link the plurality of attributes to the plurality of response templates comprises instructions to:

form a megacategory, with the megacategory linking a combination of attributes to one of the plurality of response templates using one of the plurality of Boolean expressions.

43. The apparatus of claim 40 wherein being equivalent comprises being semantically equivalent.

44. The apparatus of claim 40, further comprising instructions to:

store the plurality of attributes and the plurality of response templates.

45. The apparatus of claim 42, wherein instructions to perform an initialization comprise instructions to:

convert an AIML category having one pattern and one response template to an equivalent combination of one attribute and one megacategory, the one megacategory comprising the one response template; and
store the equivalent combination.

46. The apparatus of claim 42 wherein instructions to perform an initialization comprise instructions to:

expand the patterns;
arrange the patterns in alphabetical order;
assign a specificity rank to each pattern; and
add each pattern to an attribute trie.

47. The apparatus of claim 46 wherein instructions to perform an initialization comprise instructions to:

expand "that" patterns;
arrange "that" patterns alphabetically;

15

assign a specificity rank to each “that” pattern; and
add each “that” pattern to a “that” trie.

48. The apparatus of claim 47 wherein instructions to perform an initialization comprise instructions to:

expand megacategories into conjunctions;
order the conjunctions alphabetically; and
add each megacategory to a megacategory trie.

49. The apparatus of claim 48, wherein instructions to determine an appropriate response template comprise instructions to:

divide the input into word segments;
traverse the attribute trie with the input to determine matching attributes;
traverse the “that” trie to determine matching “that” patterns from a previous output; and
traverse the megacategory trie with matching attributes patterns and matching “that” patterns to determine matching megacategories.

50. The apparatus of claim 49 wherein instructions to traverse the attribute trie comprise instructions to:

assign a variable to each pattern; and
set the variable to true for each pattern that matches a portion of the input.

16

51. The apparatus of claim 49, further comprising instructions to:

determine that one megacategory matches, and when there is a determined match, select a response template corresponding to the one megacategory.

52. The apparatus of claim 49, further comprising instructions to:

determine that more than one megacategory matches, and when there are determined matches,
select a response template corresponding to a megacategory having the highest summed specificity rank from the set of patterns.

53. The apparatus of claim 52, further comprising instructions to:

determine that more than one megacategory has the highest summed specificity rank, select a megacategory having a pattern having the single highest specificity rank.

54. The apparatus of claim 49, further comprising instructions to:

determine that no megacategories match, add matched attributes from a last user input to the current matched attributes; and
determine a matching megacategory.

55. The apparatus of claim 40, further comprising instructions to:

execute the appropriate response template.

* * * * *